Lite-Fi Driver
Usage and Integration in Linux

Version 1.0
September 23rd 2011

Redpine Signals, Inc.
2107 N. First Street, #680
San Jose, CA 95131.
Tel: (408) 748-3385
Fax: (408) 705-2019
Email: info@redpinesignals.com
Website: www.redpinesignals.com

Disclaimer:

The information in this document pertains to information related to Redpine Signals Inc. products. This information is provided as a service to our customers, and may be used for information purposes only.

Redpine assumes no liabilities or responsibilities for errors or omissions in this document. This document may be changed at any time at Redpine's sole discretion without any prior notice to anyone. Redpine is not committed to updating these documents in the future.

Copyright © 2011 Redpine Signals, Inc. All rights reserved.

Table of Contents

1: About This Guide 4
2: Lite-Fi Host Driver – Setup Requirements 5
2.1: Hardware Requirements 5
2.2: Software Requirements for Linux 5
3: Using SDIO in Linux-2.6.30 5
4: Building the Lite-Fi Driver 6
5: Installing the Lite-Fi Driver 7
6: Uninstalling the Lite-Fi Driver 8
7: Using Wi-Fi Client Driver 8
8: Wireless Tools for Linux 9
9: Using Wi-Fi in an Application 9
10: Appendix 11
10.1: Sample Client Program Reading from Socket 11

1: About This Guide

The purpose of this document is to provide details of Redpine's Lite-Fi® Wi-Fi® client driver and its integration with Linux operating system. A user should be able to use Wi-Fi in their application with the help of this driver. The description is valid for Lite-Fi family devices such as RS9110-N-11-02 and RS9110-N-11-03 for installing the Wi-Fi driver under Linux platform. The Wi-Fi station driver is used to connect to an Access Point and carry out some data transfer tests using Lite-Fi family devices.

This installation guide covers the following topics:

- Lite-Fi Host driver – setup requirements
- Building the Lite-Fi driver
- Installing the Lite-Fi driver
- Uninstalling the Lite-Fi driver
- Using Lite-Fi driver
- Description of wireless tools on Linux
- Using Wi-Fi in sample client application

2: Lite-Fi Host Driver – Setup Requirements

This section describes the hardware and software environment for the installation and operation of the Redpine's Lite-Fi RS9110-N-11-02 and RS9110-N-11-03 based evaluation boards. The following configuration is required to install and use the Lite-Fi SDIO host driver on Linux platforms.

2.1: Hardware Requirements

- A Microprocessor development platform with SDIO slot
    - A microprocessor can be Atmel SAM9 platform, Freescale i.MX platform, Nvidia Tegra 2 and many other similar microprocessors
- Lite-Fi RS9110-N-11-02 or RS9110-N-11-03 Evaluation Board

2.2: Software Requirements for Linux

- Fedora Core 10 and above - Linux [1] operating system
- Linux kernel source code with SDIO stack enabled
    - Please refer to section 'Using SDIO in Linux-2.6.30' for details.

3: Using SDIO in Linux-2.6.30

The Lite-Fi host driver uses the SDIO stack available in Linux-2.6.30 kernel. To configure the SDIO stack in Linux-2.6.30, you need the kernel source code for the Kernel version 2.6.30. The kernel source can be obtained from http://www.kernel.org/

It has been noted on X86, the DMA is enabled in SDIO host controller, which causes instability issues at SDIO host controller level. So, the following lines in function *sdhci_add_host* of the file *linux-2.6.30/drivers/mmc/host/sdhci.c*, should be commented.

```
If ( host->quirks & SDHCI_QUICK_FORCE_DMA)
        host->flags |= SDHCI_USE_DMA;
else if ( ! (caps & SDHCI_CAN_DO_DMA))
        DBG("controller doesn't have DMA capability\n");
else
        host->flags |= SDHCI_USE_DMA;
```

Now, configure the kernel's features as given below:

- #cd linux-2.6.30

[1] Redpine Signals has performed full functional testing of the driver with Linux kernel 2.6.30.

- Run 'make menuconfig' command and make sure that the following features are enabled in the menu.
    - Select <Device Drivers > and press Enter.
    - Press 'M' to modularize the 'MMC/SD/SDIO card support' feature
    - Select <MMC/SD/SDIO card support > and press Enter.
    - Press 'M' to modularize the following in the 'MMC/SD/SDIO card support" submenu:
      <M>Secure Digital Host Controller Interface Support.
      <M>SDHCI Support on PCI bus.
    - Press 'Exit' button till you get back to the main menu

- Now, exit from the main menu and save the kernel configuration.
- Run 'make' command. This builds the customized kernel.
- Run 'make modules_install' command.
- Run 'make install' command. This ensures that the customized kernel is installed and the boot loader is updated appropriately.
- Run 'reboot' command.
- Select the Linux 2.6.30 kernel. Once you have successfully booted. The kernel is now ready for use.

4: Building the Lite-Fi Driver

The Lite-Fi driver is released in .tgz format

1. Example: RS.GENR.LNX.SD.X.Y.Z.tgz.
2. Copy the file into any directory in Linux system. For example, into /work/home/paul.
3. Follow the instructions given below to extract the driver source.
    - `#cd /work/home/paul`
    - `#tar zxvf RS.GENR.LNX.SD.X.Y.Z.tgz`
4. A directory `RS.GENR.LNX.SD.X.Y.Z` should be created.
5. Change the current working directory to above created directory using:
    - #cd `RS.GENR.LNX.SD.X.Y.Z`
6. You should see two more .tgz files in this directory:

    `RS.GENR.LNX.SD.X.Y.Z_NON_GPL.tgz`

    `RS.GENR.LNX.SD.X.Y.Z_GPL.tgz`
7. Extract the above two files
    - `#tar zxvf RS.GENR.LNX.SD.X.Y.Z_NON_GPL.tgz`

- `#tar zxvf RS.GENR.LNX.SD.X.Y.Z_GPL.tgz`

8. Two directories with above name should be created

   Note: The Lite-Fi driver comprises of two drivers namely the client driver and the master driver. The folder RS.GENR.LNX.SD.X.Y.Z_NON_GPL contains the client driver source and the folder RS.GENR.LNX.SD.X.Y.Z_GPL contains the master driver source.

9. Run the following commands, in the same order, to build the Lite-Fi driver:
10. To build the client module run the following commands:
    - `#cd RS.GENR.LNX.SD.X.Y.Z_NON_GPL/OSD/LINUX/client`
    - `#make clean`
    - `#make`
11. To build the master module run the following commands:
    - `#cd ../../../../RS.GENR.LNX.SD.X.Y.Z_GPL/OSD/LINUX/master`
    - `#make clean`
    - `#make`
12. Execute the following to verify that the build process has been completed successfully:
    - `#cd ../release`
    - `#sh build_check.sh`

    Note: Upon successful compilation, a message, "Lite-Fi driver is built/compiled successfully" appears on the console.

    Note: You may need to change the KERNELDIR macro in the 'Makefile' of the GPL/Non-GPL directories to point to the kernel sources on the system if the default path as specified by this macro is different for your system.

5: Installing the Lite-Fi Driver

To load the host driver module, the following steps should be followed:

- Login as a super user

  Make sure that you are in 'release' folder under 'RS.GENR.LNX.SD.X.Y.Z_GPL/OSD/LINUX/release' directory.
- #cd RS.GENR.LNX.SD.X.Y.Z/RS.GENR.LNX.SD.X.Y.Z_GPL/OSD/LINUX/release
- Run the following script to install the Lite-Fi driver.

  #sh install_driver.sh

Once the driver is successfully loaded, a message, "Lite-Fi driver is installed successfully" appears on the screen. Otherwise, follow the instructions as on the screen.

- Insert the Lite-Fi SD card into the appropriate SDIO slot, and run the following script to verify that the card is detected correctly.

  #sh detection_check.sh

  If the card has been detected, a message, "The Lite-Fi card is detected successfully" appears on the screen. Otherwise, follow the instructions as on the screen.

6: Uninstalling the Lite-Fi Driver

For hot plug removal of the Lite-Fi card from the SDIO slot, you need to perform the following steps:

- Move to 'release' folder found under 'RS.GENR.LNX.SD.X.Y.Z_GPL/OSD/LINUX' directory
- #cd RS.GENR.LNX.SD.X.Y.Z/RS.GENR.LNX.SD.X.Y.Z_GPL/OSD/LINUX/release
- Run following script.

  #sh uninstall_driver.sh

The script will stop active transmit and receive, if any, and unload the LiteFi™ driver. At this point, a message "Lite-Fi driver is uninstalled successfully" appears on the screen.

Then, you can safely remove the Lite-Fi card from the system.

7: Using Wi-Fi Client Driver

The driver compilation procedure described in this document enabled the developer to create a driver module (.ko) files for Linux OS.
The commands are below

- insmod rsi_client.ko
- insmod rsi_master.ko driver_mode=1 firmware_path="release/"
  - driver_mode = 1, loads the driver in regular Wi-Fi client mode
  - driver_mode = 2, loads the driver in PER Wi-Fi Test mode (used only for special cases.
  - firmware_path argument specifies the location of firmware to be loaded on Redpine Wi-Fi module

The wpa_supplicant binary is used to scan, configure and connect to a Wi-Fi network. The supplicant requires the interface name (wlan0) to be provided as an argument. The configuration settings for wpa_supplicant are provided in a wifi_settings.cfg file.

The Wi-Fi client uses SDIO interface between the Wi-Fi module and the Host processor. The driver depends on the SDIO host controller to communicate with Host processor for commands and configuration items. It becomes imperative to reset the SDIO interface also when the Wi-Fi module is has to be powered off or reset.

The installation and un-installation scripts provided perform the exact same function by resetting the SDIO stack. This procedure does assume that SDIO Stack is also loaded as a module in Linux Kernel.

Reference commands

- rmmod sdhci-pci
- rmmod sdhci
- rmmod mmc_block
- rmmod mmc_core

8: Wireless Tools for Linux

Wireless Extensions (WE) (A.K.A tools) for Linux are provided by Linux distributions as a set of commands in the operating system. A few sample commands for the extensions are

- iwconfig
- iwevent
- iwlist
- iwpriv

The detailed description of individual commands support by Redpine Signals is available in "LiteFi Driver User Manual" document.

Repdine Signals Wi-Fi client driver interfaces with the Linux Wireless Extensions (WE). The extensions are generic API's which can be used by in user space for configuration and measurement of WLAN parameters.

9: Using Wi-Fi in an Application

The developers can use regular TCP/IP and UDP socket programming API's of Linux standard libraries to use the wireless network interface.

A sample client socket program is provided in Appendix 10.1:

Notice that there is no interface (eth0 or wlan0) specified in the program. The network interface [2] to be used for sending and receiving packets in an application program is determined by the routing table. Refer to "iproute" and "ifconfig" commands on Linux. The OS will route the packet from the application to the relevant networking interface (eth0, wlan0) based on the IP address of the destination packet specified in the program.

[2] Redpine's interface in Linux is typically WLAN0 – Wireless LAN interface

Each network interface is typically on a different internet subnet. The OS routing table determines the packet sending and reception from a network interface based on the IP address specified.

10: Appendix

10.1: Sample Client Program Reading from Socket

```
/* simple-client.c
 *
 * Copyright (c) 2000 Sean Walton and Macmillan Publishers.  Use may be in
 * whole or in part in accordance to the General Public License (GPL).
 *
 * THIS SOFTWARE IS PROVIDED BY THE REGENTS AND CONTRIBUTORS ``AS IS'' AND
 * ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
 * ARE DISCLAIMED.  IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE
 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
 * OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
 * HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
 * LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
 * OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
 * SUCH DAMAGE.
*/

/*************************************************************/
/*** simple-client.c                                       ***/
/***                                                       ***/
/*************************************************************/

/*************************************************************
*   This is a simple client socket reader.  It opens a socket, connects
*   to a server, reads the message, and closes.
*************************************************************/
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <sys/socket.h>
#include <resolv.h>

#define PORT_TIME      13          /* "time" (not available on RedHat) */
#define PORT_FTP       21          /* FTP connection port */

/* IP Address of Remote Server connected using Wi-Fi network */
#define SERVER_ADDR    "192.168.0.1"

#define MAXBUF         1024

int main()
{   int sockfd;
    struct sockaddr_in dest;
    char buffer[MAXBUF];

    /*---Open socket for streaming---*/
    if ( (sockfd = socket(AF_INET, SOCK_STREAM, 0)) < 0 )
```

```
    {
       perror("Socket");
       exit(errno);
    }

    /*---Initialize server address/port struct---*/
    bzero(&dest, sizeof(dest));
    dest.sin_family = AF_INET;
    dest.sin_port = htons(PORT_FTP);
    if ( inet_aton(SERVER_ADDR, &dest.sin_addr.s_addr) == 0 )
    {
       perror(SERVER_ADDR);
       exit(errno);
    }

    /*---Connect to server---*/
    if ( connect(sockfd, (struct sockaddr*)&dest, sizeof(dest)) != 0 )
    {
       perror("Connect ");
       exit(errno);
    }

    /*---Get "Hello?"---*/
    bzero(buffer, MAXBUF);
    recv(sockfd, buffer, sizeof(buffer), 0);
    printf("%s", buffer);

    /*---Clean up---*/
    close(sockfd);
    return 0;
}
```

*****